# Learnable Evolutionary Search across Heterogeneous Problems via Kernelized Autoencoding

Lei Zhou, Liang Feng*, Abhishek Gupta, Yew-Soon Ong, *Fellow, IEEE*

***Abstract*—The design of evolutionary algorithm with learning capability from past search experiences has attracted growing research interests in recent years. It has been demonstrated that the knowledge embedded in the past search experience can greatly speed up the evolutionary process if properly harnessed. Autoencoding evolutionary search (AEES) is a recently proposed search paradigm which employs a single-layer denoising autoencoder to build the mapping between two problems by configuring the solutions of each problem as the input and output for the autoencoder, respectively. The learned mapping makes it possible to perform knowledge transfer across heterogeneous problem domains with diverse properties. It has shown a promising performance of learning and transferring the knowledge from past search experiences to facilitate the evolutionary search on a variety of optimization problems. However, despite the success enjoyed by AEES, the linear autoencoding model cannot capture the nonlinear relationship between the solution sets used in the mapping construction. Taking this cue, in this paper, we devise a kernelized autoencoding to construct the mapping in a Reproducing Kernel Hilbert Space (RKHS), where the nonlinearity among problem solutions can be captured easily. Importantly, the proposed kernelized autoencoding also holds a closed-form solution which will not bring much computational burden in the evolutionary search. Furthermore, a kernelized autoencoding evolutionary search paradigm (KAES) is proposed that adaptively selects the linear and kernelized autoencoding along the search process in pursuit of effective knowledge transfer across problem domains. To validate the efficacy of the proposed KAES, comprehensive empirical studies on both benchmark multiobjective optimization problems as well as real-world vehicle crashworthiness design problem are presented.**

***Index Terms*—Evolutionary optimization, knowledge transfer, kernelization, nonlinear.**

## I. Introduction

Evolutionary algorithm (EA) is a popular population-based search method for solving complex optimization problems, where derivatives may not be available or multiple local optima may exist, that are intractable by conventional mathematical methods [1], [2]. Due to its strong search ability and easy implementation, EA has been successfully applied to solve a variety of optimization problems, including continuous optimization [3], [4], discrete optimization [5], [6], single objective optimization (SOO) [7], [8] and multi/many-objective optimization (MOO/MaOO) [9–11] as well as robust optimization [12], [13] and dynamic optimization [14], [15] in uncertain environments, etc.

Over the years, a number of research efforts have been made on developing advanced EAs in the literature, many of which focus on the design of efficient evolution mechanisms, involving new genetic operators [16], [17], reliable selection criteria [18], [19] as well as intensive local search heuristics [20], [21]. It is noted that most existing EAs conduct the optimization from scratch without considering search experiences on past solved problems which could be used to speed up the evolutionary process [22], [23]. Although similar ideas have been well established in the realm of machine learning, where it is referred to as "Transfer Learning" [24], [25], relevant research in evolutionary optimization literature is still in its infancy. As problems seldom exist in isolation, related problems usually contain useful information that could enhance the optimization process across problem domains. Therefore, there is a growing interest in improving evolutionary search performance by designing knowledge transfer components that can leverage the useful traits found in solving one problem to help another.

The early attempts on knowledge transfer in EA were to archive high-quality solutions of past-solved problems, and directly reuse these solutions when related problems are encountered. For instance, Louis *et al.* [26] proposed a genetic algorithm equipped with a case-based memory of the optimized solutions of past problems. Rather than starting anew on each problem, the archived solutions from similar problems are periodically injected into the population to accelerate the optimization process of a genetic algorithm via case-based reasoning. In [27], a performance boost is observed when addressing a traveling salesman problem by injecting the solutions of previously solved similar problems into the initial population of the genetic algorithm. Similar ideas are also widely adopted in dynamic optimization, where useful information or solutions from the current environment are stored and reused later in new environments. Representative examples can be found in [28–30]. However, as these methods rely on the direct reuse of past solutions, they may fail on problems with different ranges or dimensionality of search space.

In recent years, researchers have started exploring flexible yet efficient ways to transfer knowledge across problem domains with distinct properties. For instance, a memetic

Lei Zhou and Yew-Soon Ong are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. Email: {lei.zhou, asysong}@ntu.edu.sg.

Liang Feng is with the College of Computer Science, Chongqing University, China. Email: liangf@cqu.edu.cn.

Abhishek Gupta is with Singapore Institute of Manufacturing Technology (SIMTech), A*STAR, Singapore. Email: abhishek_gupta@simtech.astar.edu.sg.

* Liang Feng is the corresponding author (email: liangf@cqu.edu.cn).

algorithm is designed in [31] to transfer the structured knowledge in the form of a transformation matrix learned from previous problem-solving experiences across the domains of vehicle routing and arc routing. Gupta *et al.* [32] presented a multifactorial evolutionary algorithm to optimize multiple optimization tasks simultaneously, in which the knowledge of the problems is exchanged implicitly within a unified search space. In [33], multiple base surrogate models are stacked using a meta-regression method as an alternative way of knowledge sharing among distinct but related sources. More recently, Da *et al.* [34] proposed to encode the knowledge acquired from past optimization exercises in the form of probabilistic models. Elite solutions can then be adaptively sampled from the model to facilitate a target search process.

Autoencoding evolutionary search (AEES) is a recently proposed search paradigm using a single-layer denoising autoencoder (DAE) to build the mapping between two problems by configuring the solutions of each problem as the input and output for the autoencoder, respectively [35]. The learned mapping makes it possible to transfer the high-quality solutions found in one problem to another problem domain, even though the two domains possess different problem properties, such as dimensionality, range of decision variables and number of objectives. The efficacy of AEES has been verified on both commonly used multiobjective optimization problems and real world applications. However, despite the success of AEES, it is worth noting that its linearly formulated DAE can only model the linear relationship among problem solutions, which may affect the performance of AEES when the solutions are nonlinearly correlated. In addition, AEES transfers the whole optimized population from one problem to the other, which is inefficient in cases where similar solutions exist in the population.

Keeping the above in mind, in this paper, we propose a new learnable evolutionary search paradigm, namely kernelized autoencoding evolutionary search (KAES). In KAES, a kernelized autoencoding method is derived to conduct kernelization on a single-layer DAE for building the mapping across heterogeneous problems in a Reproducing Kernel Hilbert Space (RKHS), where the nonlinearity between the solution sets used in the mapping construction can be captured easily. As a result of the kernel trick, the newly-derived nonlinear mapping holds a closed-form solution so that it will not bring much additional computational burden. Rather than solely use the kernelized autoencoding or the linear autoencoding as in [35], KAES combines both kinds of autoencoding and adaptively determine which one to select based on the similarity of the two solution sets measured by Kullback-Leibler divergence [36]. With the learned mapping, the knowledge from past search experience can be transferred to the current problem in the form of adapted problem solutions. Further, the solutions for the mapping construction are sorted by solution quality in order to introduce a high ordinal correlation towards effective knowledge transfer. Lastly, in contrast to [35] that reuses the whole population of the past optimized solutions, the proposed KAES transfers a limited number of representative elite solutions which greatly improves the computational efficiency. Comprehensive empirical studies have been conducted on 12 commonly used multiobjective benchmarks [37, 38] as well as a real-world case study on the vehicle crashworthiness design problem in the automotive industry [39]. The experimental results confirm the efficacy of the proposed kernelized autoencoding evolutionary search paradigm.

The rest of this paper is organized as follows. Section II briefly introduces the autoencoding evolutionary search as well as the motivation of this work. Subsequently, the details of the proposed kenelized autoencoding method and evolutionary search paradigm are elaborated in Section III. Section IV provides the comprehensive empirical studies on the commonly-used multiobjective benchmark problems. Further, a real-world case study on the vehicle crashworthiness design problem is presented in Section V. Lastly, the concluding remarks of this paper are drawn in Section VI.

## II. Preliminaries

In this section, we first give a brief introduction of the autoencoding evolutionary search. Next, the motivation of this work is demonstrated with an illustrative example.

### A. Autoencoding Evolutionary Search

Denoising autoencoder is a specific form of neural network that attempts to reconstruct the initial input from its corrupted version [40]. It has been successfully applied for solving many challenging learning problems, such as domain adaptation [41, 42], natural language processing [43, 44], sentiment analysis [45, 46], etc.

Formally, given the input vector $\mathbf{x} \in [0,1]^d$, an denoising autoencoder first corrupts $\mathbf{x}$ into $\tilde{\mathbf{x}}$ before mapping it to a hidden representation $\mathbf{y} \in [0,1]^{d'}$ via $\mathbf{y} = s(\mathbf{W}\mathbf{x} + \mathbf{b})$, where $\mathbf{W}$ and $\mathbf{b}$ are the weight and bias between the input and the hidden layers, respectively, and $s$ is the sigmoid activation function, i.e., $s(\mathbf{x}) = [1/(1+e^{-\mathbf{x}})]$. Next, $\mathbf{y}$ is decoded as $\mathbf{z} = s(\mathbf{W}'\mathbf{y} + \mathbf{b}') \in [0,1]^d$ with the aim to reconstruct $\mathbf{x}$ by minimizing the reconstruction error as follows:

$$\min_{\mathbf{W},\mathbf{W}',\mathbf{b},\mathbf{b}'} = \frac{1}{n}\sum_{i=1}^{n} L(\mathbf{x}_i, \mathbf{z}_i), \tag{1}$$

where $n$ denotes the number of data instances, and $L$ is a loss function, such as the square loss, Kullback-Leibler divergence, etc.

Inspired by the fact that the hidden representation $\mathbf{y}$ actually builds a connection between the corrupted inputs $\tilde{\mathbf{x}}$ and the initial input $\mathbf{x}$, autoencoding evolutionary search is proposed in [35] to leverage a single layer denoising autoencoder to construct the mapping across problem domains by treating the solutions of one optimization problem as the corrupted version of the solutions of the other problem. Particularly, as depicted in Fig. 1, let $\mathbf{P} \in \mathcal{R}^{N\times d}$ and $\mathbf{Q} \in \mathcal{R}^{N\times d}$ represent the solution set of two different optimization problems $T_1$ and $T_2$, respectively, i.e., $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ and $\mathbf{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_N\}$, where $N$ denotes the number of solutions in each set and $d$ is the solution's dimension[1].

[1]If p and q have different dimensions, we pad p or q with zeros to make them be of equal dimensionality.

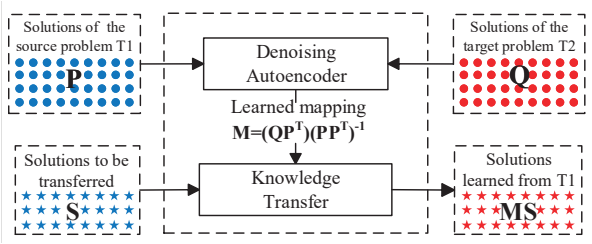



Fig. 1. Illustration of the autoencoding-based knowledge transfer.

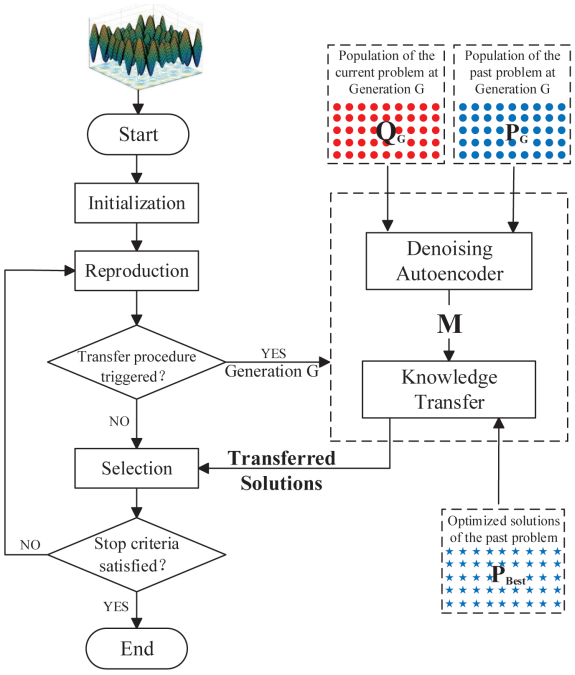



Fig. 2. Workflow of the autoencoding evolutionary search paradigm.

The mapping from $T_1$ to $T_2$ can be naturally built through a denoising autoencoder by using $\mathbf{P}$ as the input and $\mathbf{Q}$ as the output accordingly. Further, the corrupted input is reconstructed with a single level mapping $\mathbf{M}:\mathcal{R}^d \rightarrow \mathcal{R}^d$, that minimizes the squared reconstruction loss as follows:

$$\mathcal{L}_{sq} = \frac{1}{2N}\sum_{i=1}^{N}\|\mathbf{q}_i - \mathbf{M}\mathbf{p}_i\|^2. \quad (2)$$

The solution of Eq. (2) can be expressed as the well-known closed-form solution for ordinary least squares [47], which is given by:

$$\mathbf{M} = (\mathbf{Q}\mathbf{P}^T)(\mathbf{P}\mathbf{P}^T)^{-1}. \quad (3)$$

The learned $\mathbf{M}$ thus builds a mapping from $T_1$ to $T_2$. The solutions that are transferred from $T_1$ to $T_2$ can be generated simply by multiplying with $\mathbf{M}$.

With the autoencoding based knowledge transfer component integrated into the EA framework, the workflow of the autoencoding evolutionary search paradigm is depicted in Fig. 2. As shown in Fig. 2, a transfer procedure is triggered periodically during the evolution process. The current population and the archived population of a past optimized problem at generation $G$ are used to construct the mapping $\mathbf{M}$. Subsequently, the transferred solutions, which are obtained by multiplying the optimized solutions of the past problem with $\mathbf{M}$, are injected into the current population. Lastly, the solutions go through the process of natural selection to automatically alleviate the negative transfer.

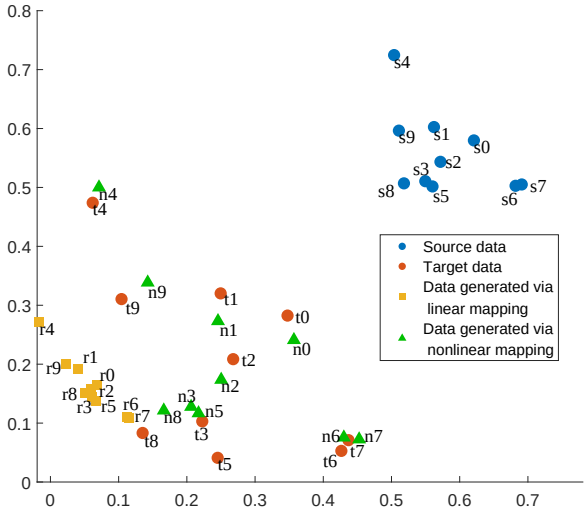



Fig. 3. Nonlinear mapping outperforms linear mapping in reconstructing dataset $\mathbf{T}=\{\mathbf{t}_0,\ldots,\mathbf{t}_9\}$ from nonlinear correlated dataset $\mathbf{S}=\{\mathbf{s}_0,\ldots,\mathbf{s}_9\}$.

### B. Motivation

Although the autoencoding evolutionary search has demonstrated its superiority on both benchmark problems and real-world applications [35], it cannot capture the potentially nonlinear relationship between datasets with a linear autoencoding model, which may degrade the mapping accuracy and thus discount the potential improvement.

As an illustrative example, Fig. 3 shows the results of the linear and nonlinear mapping in reconstructing one dataset from the other, where the two datasets have nonlinear correlation. In Fig. 3, $\mathbf{S} = \{\mathbf{s}_0,\ldots,\mathbf{s}_9\}$ and $\mathbf{T} = \{\mathbf{t}_0,\ldots,\mathbf{t}_9\}$ are two sets of data, where $\mathbf{t}_i = \phi(\mathbf{s}_i)$ is generated from $\mathbf{s}_i$ via a nonlinear function $\phi$. Particularly, the first 8 samples in $\mathbf{S}$ and $\mathbf{T}$, i.e., $\mathbf{S_m} = \{\mathbf{s}_0,\ldots,\mathbf{s}_7\}$ and $\mathbf{T_m} = \{\mathbf{t}_0,\ldots,\mathbf{t}_7\}$, are used as training data to build the linear mapping $\mathbf{M}_L$ and nonlinear mapping $\mathbf{M}_N$, while the last two samples $\mathbf{S_v} = \{\mathbf{s}_8,\mathbf{s}_9\}$ and $\mathbf{T_v} = \{\mathbf{t}_8,\mathbf{t}_9\}$ are employed as validation data. $\mathbf{R_m} = \{\mathbf{r}_0,\ldots,\mathbf{r}_7\}$ and $\mathbf{N_m} = \{\mathbf{n}_0,\ldots,\mathbf{n}_7\}$ are generated via $\mathbf{M}_L$ and $\mathbf{M}_N$, respectively, to reconstruct $\mathbf{T_m}$ from $\mathbf{S_m}$. As can be observed, $\mathbf{N_m}$ achieved better approximation of the distribution of $\mathbf{T_m}$ than $\mathbf{R_m}$. For instance, $\mathbf{n}_1$, $\mathbf{n}_3$ and $\mathbf{n}_5$ locate close to $\mathbf{t}_1$, $\mathbf{t}_3$ and $\mathbf{t}_5$ respectively, while $\mathbf{r}_1$, $\mathbf{r}_3$ and $\mathbf{r}_5$ are far away. Similar observation can also be obtained on the validation set $\mathbf{S_v}$ and $\mathbf{T_v}$, which indicates that the nonlinear mapping can not only fit well into the training data, but also have a good generalization to the unseen data from the same distribution. From the above, it can be concluded that the nonlinear mapping is able to capture the nonlinearity among data which leads to a higher mapping accuracy than the linear mapping.

A straightforward way to extend the linear autoencoding model with nonlinearity-handling capability is to include additional hidden layers with nonlinear activation functions. In this regard, many state-of-art autoencoder models proposed in the literature, such as contractive autoencoder and robust autoencoder [48], are ready for use. Despite the powerful learning ability of these models, they usually require iterative procedures, e.g. stochastic gradient descent, to learn the model parameters, which is computationally expensive. The architecture of the model also has a great effect on its performance. The optimal architecture is usually problem-dependent and the hyper-parameters are exhaustive to tune. As the AEES paradigm keeps updating the inter-task mapping throughout the search process, the large amount of computational time spent on the training of the multilayer autoencoder models is undesirable for the optimization. Consequently, a computationally efficient nonlinear autoencoding method is urgently required.

Taking this cue, towards an enhanced representation ability and transfer performance, we thus propose a kernerlized autoencoding method to efficiently build a nonlinear mapping across heterogeneous problems, which will be elaborated in the next section.

## III. Evolutionary search across Heterogeneous Problems via Kernelized Autoencoding

In this section, we first give the derivation of the proposed kernelized autoencoding method, including the mapping construction and the generation of the transferred solutions. Next, the self-adaption process between linear and nonlinear mapping is introduced. Lastly, the details of the proposed kernelized autoencoding evolutionary search paradigm is presented.

### A. The proposed kernelized autoencoding

As aforementioned, the linear autoencoding method adopted in [35] may lead to an inaccurate mapping since it can only model linear relationship between two datasets. In machine learning, it is a common practice to handle nonlinearity by mapping the data into a high-dimensional feature space based on the kernel method. The use of kernel function makes it possible to manipulate the data in the high-dimensional space at a low cost. By applying kernelization into the learning of $\mathbf{M}$, we propose a kernelized autoencoding method which has the capability to capture the nonlinearity between datasets.

To be specific, suppose the solution set $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ in Eq. 2 is mapped to a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ by a nonlinear mapping function $\Phi$, Eq. 3 is then reconstructed as:

$$\mathcal{L}_{sq}(\mathbf{M}) = \frac{1}{2N}\sum_{i=1}^{N}\|\mathbf{q}_i - \mathbf{M}\Phi(\mathbf{p}_i)\|^2. \tag{4}$$

Given $\mathbf{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_N\}$, Eq. 4 can be reduced to the matrix form as follows:

$$\mathcal{L}_{sq}(\mathbf{M}) = \frac{1}{2N}tr[(\mathbf{Q} - \mathbf{M}\Phi(\mathbf{P}))^{\mathrm{T}}(\mathbf{Q} - \mathbf{M}\Phi(\mathbf{P}))], \tag{5}$$

where *tr*$(\cdot)$ denotes the trace of a matrix. According to [49], the linear mapping $\mathbf{M}$ in $\mathcal{H}$ can be represented as a linear combination of the data points $\Phi(\mathbf{X})$ in $\mathcal{H}$, that is, $\mathbf{M} = \mathbf{M}_k\Phi(\mathbf{X})^{\mathrm{T}}$. Therefore, the term $\mathbf{M}\Phi(\mathbf{P})$ in Eq. 5 can be rewritten as $\mathbf{M}_k\Phi(\mathbf{P})^{\mathrm{T}}\Phi(\mathbf{P})$. Denote the kernel matrix as $\mathbf{K}(\mathbf{P},\mathbf{P}) = \Phi(\mathbf{P})^{\mathrm{T}}\Phi(\mathbf{P})$. The (*i, j*)th element of $\mathbf{K}(\mathbf{P},\mathbf{P})$ is $\kappa(\mathbf{p}_i, \mathbf{p}_j)$, where $\kappa(\cdot,\cdot)$ is the kernel function. Then, the objective function becomes:

$$\mathcal{L}_{sq}(\mathbf{M}_k) = \frac{1}{2N}tr[(\mathbf{Q} - \mathbf{M}_k\mathbf{K}(\mathbf{P},\mathbf{P}))^{\mathrm{T}}(\mathbf{Q} - \mathbf{M}_k\mathbf{K}(\mathbf{P},\mathbf{P}))]. \tag{6}$$

Finally, Eq. 6 can be deduced into a closed-form solution as:

$$\mathbf{M}_k = \mathbf{Q}\mathbf{K}(\mathbf{P},\mathbf{P})^{\mathrm{T}}(\mathbf{K}(\mathbf{P},\mathbf{P})\mathbf{K}(\mathbf{P},\mathbf{P})^{\mathrm{T}})^{-1}. \tag{7}$$

To transfer a set $\mathbf{S}$ of elite solutions from one problem to the other, $\Phi$ is first performed on $\mathbf{S}$ to get $\Phi(\mathbf{S})$, which is then multiplied with $\mathbf{M}$. Thus, the transferred solutions are obtained as follows:

$$\mathbf{TS} = \mathbf{M}\Phi(\mathbf{S}) = \mathbf{M}_k\mathbf{K}(\mathbf{P},\mathbf{S}). \tag{8}$$

Last but not the least, it is worth noting that the learning of $\mathbf{M}_k$ holds a closed-form solution in the proposed kernelized autoencoding method. Compared to the linear autoencoding, the additional cost is the calculation of the kernel matrix, which thus will not bring much computational burden in the evolutionary search process.

### B. Self-adaptation between linear and nonlinear mapping

Although nonlinear mapping is more accurate as shown in Section II-B, the linear mapping can diversify the transferred solutions and is more computationally efficient. Consequently, it should be a better idea to combine both linear and nonlinear mapping and adaptively choose the appropriate one along the search process. In particular, we propose to encourage the linear mapping if the distributions of the two populations $\mathbf{P}$ and $\mathbf{Q}$ are similar. Otherwise, the probability of nonlinear mapping is augmented so that the mapping accuracy can be guaranteed.

In this work, we adopt multivariate Gaussian probabilistic model to approximate distributions of $\mathbf{P}$ and $\mathbf{Q}$ and measure the similarity of $\mathbf{P}$ and $\mathbf{Q}$ by the Kullback-Leibler divergence (*KLD*) [36], which can be calculated by:

$$\begin{aligned} KLD(\mathbf{P}||\mathbf{Q}) =& \frac{1}{2}(tr(\Sigma_P^{-1}\Sigma_Q) + (\mu_P - \mu_Q)^T\Sigma_P^{-1}(\mu_P - \mu_Q) \\ & - D + ln(\frac{det(\Sigma_P)}{det(\Sigma_Q)})), \end{aligned} \tag{9}$$

where $tr(\cdot)$ and $det(\cdot)$ are the trace and determinant of a matrix, respectively; $\Sigma$ represents the covariance matrix and $\mu$ represents the mean vector. It is noted that before the calculation of *KLD*, solutions in $\mathbf{P}$ and $\mathbf{Q}$ are truncated into the first *D*=min($D_P$, $D_Q$) dimensions in case that the dimensionality $D_P$ of $\mathbf{P}$ and $D_Q$ of $\mathbf{Q}$ are unequal.

Then, the probability of linear mapping is calculated by:

$$prob = max(0, 1 - \frac{\sqrt{KLD(\mathbf{P}||\mathbf{Q})}}{2 * D}), \tag{10}$$

where *D* serves as a scaling factor to adaptively reduce the bias caused by dimensionality in calculating the probability

of conducting linear mapping. The probability of nonlinear mapping is thus 1-$prob$.

It can be observed that a small *KLD* which indicates $\mathbf{P}$ and $\mathbf{Q}$ have similar distributions leads to a high probability of linear mapping, while nonlinear mapping is performed frequently with a large *KLD*.

### C. The proposed evolutionary search via kernelized autoencoding

**Algorithm 1:** Pseudo Code of the Proposed Kernelized Autoencoding Evolutionary Search Paradigm

**Input:** *MFE*: Max function evaluations; *TG*: Transfer interval; *N*: Population size; *NT*: Number of solutions to be transferred; $\mathcal{A}$: Archive that records the optimization experience of the past solved problem.
**Output:** Optimized solutions for the current problem.

```
Generate the initial population Q_0.
Set the generation counter G = 1 and function evaluation counter FES = N.
Obtain the best NT optimized solutions BS from A.
while FNS < MFE do
  if mod(G, TG) == 0 then
    Obtain the archived population P_G at generation G from A.
    Generate a random number rand between 0 and 1.
    Calculate the KLD value of P_G and Q_G via Eq. 9.
    Calculate Prob via Eq. 10.
    if rand < Prob then
      Obtain M with P_G and Q_G via Eq. 3.
      Generate the NT transferred solutions TS by M * BS.
    else
      Obtain M_k with P_G and Q_G via Eq. 7.
      Generate the NT transferred solutions TS with BS and M_k via Eq. 8.
    end if
    FES = FES + NT.
    Perform environmental selection on the union of Q_G and TS to construct a new Q_G.
  end if
  Perform parents selection and offspring generation.
  Perform environmental selection on the union of Q_G and the offspring population.
  FES = FES + N.
  G = G + 1.
end while
Output the non-dominated solutions in the current population.
```

Algorithm 1 outlines the framework of the proposed kernelized autoencoding evolutionary search paradigm (KAES). In KAES, an archive $\mathcal{A}$ that records the population of the past-solved problem $T_2$ generation by generation is provided. $\mathcal{A}$ serves as the knowledge pool that maintains optimization experience of $T_2$ to facilitate the solving of current problem $T_1$. In the following, we denote $\mathbf{P}_G$ and $\mathbf{Q}_G$ as the population of $T_1$ and $T_2$ at generation $G$, respectively.

As can be observed from Algorithm 1, to optimize $T_1$, KAES performs initialization, reproduction and selection as same as the standard EA, except the transfer procedure activated every *TG* generations. In the transfer phase, the KL divergence of $\mathbf{P}_G$ and $\mathbf{Q}_G$ is firstly calculated to measure the similarity of the two populations, which is then used to obtain the probability of linear and nonlinear mapping. If linear mapping is selected, the connectivity matrix $\mathbf{M}$ is built via Eq. 3 and the transferred solutions $\mathbf{TS}$ is generated by multiplying $\mathbf{M}$ with previously optimized solutions $\mathbf{BS}$. Otherwise, Eq. 7 is executed to calculate the nonlinear mapping matrix $\mathbf{M}_k$ and the transferred solutions is obtained via Eq. 8. In KAES, we select the best *NT* optimized solutions rather than transfer the whole population as in [35] so that a large number of function evaluations can be saved. It is also noted that we assume $\mathbf{P}_G$ are sorted in advance when it is stored into $\mathcal{A}$. Thus, the same sorting strategy can be applied to $\mathbf{Q}_G$ to maintain a high ordinal correlation in the mapping construction. Particularly, in SOO the solutions can be sorted based on the objective value while in MOO the sorting is conducted based on the sorting strategy of the corresponding MO solver, such as the nondominated ranking and crowding distance used in NSGAII [50].

Lastly, once the *NT* transferred solutions $\mathbf{TS}$ are obtained, the environmental selection kicks in to form a new population from the union of the current population and $\mathbf{TS}$. The survived transferred solutions then participate in the reproduction, by which means the knowledge of $T_2$ is exploited to facilitate the optimization of $T_1$. The whole process repeats until a predefined number of function evaluations *MFE* is satisfied. Note that if multiple optimized problems are available, each of them will have a particular mapping $\mathbf{M}$ or $\mathbf{M}_k$ with $T_1$ and contribute *NT* transferred solutions to $T_1$.

## IV. Empirical Study

To verify the efficacy of the proposed kernelized autoencoding evolutionary search, comprehensive studies have been conducted in this section. In particular, for a fair comparison, the proposed KAES is evaluated on the complex multiobjective problems that are adopted in [35].

### A. Experimental Configuration

Twelve commonly used multiobjective problems, including ZDT1-4,6 with two objectives [37] and DTLZ1-7 with three objectives [38] are tested in the experiment. Further, the proposed paradigm equipped with the kernelized autoencoding and linear autoencoding, as well as the autoencoding evolutionary search paradigm proposed in [35] are compared in this study, where SPEA2 [51] and NSGAII [50] are employed here as the baseline MOP solvers. Denoting A as the baseline solver, i.e., NSGAII or SPEA2, the compared algorithms are thus labeled as A-KA, A-LA and A-M1/M2 hereafter, respectively. It is noted that as in [35], A-M1 and A-M2 represent the autoencoding evolutionary search using past

TABLE I

AVERAGED VALUE AND STANDARD DEVIATION OF THE HV OBTAINED BY THE SPEA2-KA, SPEA2-LA, SPEA2-M1, SPEA2-M2 AND SPEA2 ON THE 12 ZDT/DTLZ MULTIOBJECTIVE PROBLEMS ACROSS 20 INDEPENDENT RUNS WITH 2500 AND 15000 FUNCTION EVALUATIONS. (≈, + AND – DENOTE THE COMPARED ALGORITHM STATISTICALLY SIMILAR, BETTER AND WORSE THAN EMT-KA, RESPECTIVELY.)

| Problem | SPEA2-KA | | SPEA2-LA | | SPEA2-M1 | | SPEA2-M2 | | SPEA2 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | FES=2500 | FES=15000 | FES=2500 | FES=15000 | FES=2500 | FES=15000 | FES=2500 | FES=15000 | FES=2500 | FES=15000 |
| ZDT1 | **6.54E-01** (1.92E-03) | 6.56E-01 (3.41E-04) | 6.42E-01 – (1.69E-02, 1.9%) | 6.56E-01 ≈ (4.67E-04) | 5.85E-01 – (7.74E-02, 11.7%) | 6.56E-01 ≈ (5.14E-04) | 6.18E-01 – (5.56E-02, 5.8%) | 6.56E-01 ≈ (2.74E-04) | 2.07E-01 – (9.09E-04, 215.9%) | 6.54E-01 – (9.09E-04) |
| ZDT2 | **3.14E-01** (2.26E-02) | 3.23E-01 (3.37E-04) | 2.27E-01 – (9.24E-02, 38.1%) | 3.23E-01 ≈ (2.78E-04) | 8.64E-02 – (6.01E-02, 263.5%) | 3.23E-01 ≈ (3.08E-04) | 1.08E-01 – (8.01E-02, 191.7%) | 3.23E-01 ≈ (3.64E-04) | 1.20E-05 – (4.39E-03, 2.5E6%) | 3.18E-01 – (4.39E-03) |
| ZDT3 | **5.08E-01** (9.17E-03) | 5.13E-01 (2.17E-04) | 4.78E-01 – (5.16E-02, 6.1%) | 5.13E-01 ≈ (2.48E-04) | 3.94E-01 – (1.28E-01, 28.9%) | 5.13E-01 ≈ (6.37E-04) | 4.93E-01 – (1.95E-02, 3.0%) | 5.13E-01 ≈ (2.02E-04) | 2.35E-01 – (9.18E-03, 116.4%) | 5.09E-01 – (9.18E-03) |
| ZDT4 | **6.53E-01** (3.86E-03) | 6.57E-01 (1.42E-04) | 4.88E-01 – (1.07E-01, 34.0%) | 6.57E-01 + (1.37E-04) | 4.41E-01 – (8.42E-02, 48.0%) | 6.57E-01 ≈ (8.90E-05) | 4.21E-01 – (7.58E-02, 55.1%) | 6.57E-01 ≈ (1.58E-04) | 0.00E+00 – (4.31E-02, / ) | 6.07E-01 – (4.31E-02) |
| ZDT6 | **3.95E-01** (9.19E-04) | 3.96E-01 (1.32E-04) | 3.30E-01 – (8.38E-02, 19.6%) | 3.96E-01 ≈ (5.80E-05) | 3.08E-01 – (8.67E-02, 28.1%) | 3.96E-01 ≈ (8.00E-05) | 1.98E-01 – (1.42E-01, 99.7%) | 3.96E-01 ≈ (1.03E-04) | 0.00E+00 – (4.56E-03, / ) | 3.72E-01 – (4.56E-03) |
| DTLZ1 | 0.00E+00 (0.00E+00) | 3.55E-01 (3.03E-01) | 0.00E+00 ≈ (0.00E+00, / ) | 1.90E-01 ≈ (2.81E-01) | 0.00E+00 ≈ (0.00E+00, / ) | 1.25E-01 – (2.24E-01) | 0.00E+00 ≈ (0.00E+00, / ) | 1.45E-01 – (2.45E-01) | 0.00E+00 ≈ (2.90E-01, / ) | **5.42E-01** + (2.90E-01) |
| DTLZ2 | 3.12E-01 (1.26E-02) | 3.64E-01 (3.89E-03) | 2.76E-01 – (1.74E-02, 13.2%) | 3.34E-01 – (9.45E-03) | 2.92E-01 – (1.77E-02, 7.1%) | 3.61E-01 ≈ (6.37E-03) | 2.61E-01 – (1.83E-02, 19.8%) | 3.66E-01 ≈ (5.93E-03) | **3.36E-01** + (3.27E-03, -7.2%) | **3.75E-01** + (3.27E-03) |
| DTLZ3 | 0.00E+00 (0.00E+00) | 0.00E+00 (0.00E+00) | 0.00E+00 ≈ (0.00E+00, / ) | 0.00E+00 ≈ (0.00E+00) | 0.00E+00 ≈ (0.00E+00, / ) | 0.00E+00 ≈ (0.00E+00) | 0.00E+00 ≈ (0.00E+00, / ) | 0.00E+00 ≈ (0.00E+00) | 0.00E+00 ≈ (0.00E+00, / ) | 0.00E+00 ≈ (0.00E+00) |
| DTLZ4 | **2.73E-01** (5.11E-02) | 3.34E-01 (5.72E-02) | 1.78E-01 – (1.07E-01, 52.7%) | 3.34E-01 ≈ (5.69E-02) | 2.22E-01 – (5.58E-02, 22.9%) | 3.10E-01 ≈ (7.28E-02) | 1.88E-01 – (9.36E-02, 44.6%) | 2.97E-01 ≈ (7.56E-02) | 2.24E-01 ≈ (1.17E-01, 21.7%) | 2.49E-01 ≈ (1.17E-01) |
| DTLZ5 | 7.33E-02 (5.13E-03) | 9.00E-02 (4.23E-04) | 7.22E-02 ≈ (5.37E-03, 1.5%) | 9.01E-02 ≈ (3.66E-04) | 6.65E-02 – (4.74E-03, 10.3%) | 8.97E-02 ≈ (3.73E-04) | 6.41E-02 – (7.89E-03, 14.3%) | 8.96E-02 – (4.22E-04) | **7.65E-02** ≈ (4.38E-04, -4.2%) | **9.02E-02** ≈ (4.38E-04) |
| DTLZ6 | **8.32E-02** (9.47E-03) | 9.19E-02 (1.29E-04) | 1.33E-02 – (2.09E-02, 525.2%) | **9.20E-02** ≈ (1.10E-04) | 4.24E-03 – (1.35E-02, 1860.3%) | 9.19E-02 ≈ (1.94E-04) | 1.80E-03 – (4.98E-03, 4518.2%) | 9.19E-02 ≈ (1.18E-04) | 0.00E+00 – (0.00E+00, / ) | 0.00E+00 – (0.00E+00) |
| DTLZ7 | **2.71E-01** (1.21E-02) | 2.74E-01 (5.62E-03) | 2.45E-01 – (2.69E-02, 10.6%) | **2.77E-01** ≈ (3.48E-03) | 2.19E-01 – (3.27E-02, 23.8%) | 2.76E-01 ≈ (4.24E-03) | 2.17E-01 – (2.79E-02, 24.4%) | 2.76E-01 ≈ (5.32E-03) | 6.92E-03 – (7.71E-03, 3810.4%) | 2.65E-01 – (7.71E-03) |
| Mean | 2.95E-01 (1.07E-02) | 3.38E-01 (3.10E-02) | 2.46E-01 (4.41E-02) | 3.22E-01 (2.94E-02) | 2.18E-01 (4.68E-02) | 3.17E-01 (2.58E-02) | 2.14E-01 (4.38E-02) | 3.18E-01 (2.78E-02) | 9.04E-02 (1.89E-03) | 3.32E-01 (4.81E-01) |

TABLE II

AVERAGED VALUE AND STANDARD DEVIATION OF THE HV OBTAINED BY THE NGSAII-KA, NGSAI)-LA, AND NGSAII ON THE 12 ZDT/DTLZ MULTIOBJECTIVE PROBLEMS ACROSS 20 INDEPENDENT RUNS WITH 2500 AND 15000 FUNCTION EVALUATIONS. (≈, + AND – DENOTE THE COMPARED ALGORITHM STATISTICALLY SIMILAR, BETTER AND WORSE THAN EMT-KA, RESPECTIVELY.)

| Problem | NSGAII-KA | | NSGAII-LA | | NSGAII-M1 | | NSGAII-M2 | | NSGAII | |
|---|---|---|---|---|---|---|---|---|---|---|
| | FES=2500 | FES=15000 | FES=2500 | FES=15000 | FES=2500 | FES=15000 | FES=2500 | FES=15000 | FES=2500 | FES=15000 |
| ZDT1 | **6.52E-01** (7.37E-04) | 6.53E-01 (6.97E-04) | 6.48E-01 ≈ (1.55E-02, 0.6%) | **6.53E-01** ≈ (7.62E-04) | 5.13E-01 – (7.79E-02, 27.1%) | 6.52E-01 ≈ (8.73E-04) | 6.38E-01 – (1.76E-02, 2.2%) | 6.53E-01 ≈ (1.09E-03) | 2.83E-01 – (7.74E-04, 130.5%) | 6.52E-01 – (7.74E-04) |
| ZDT2 | **3.17E-01** (3.05E-03) | 3.20E-01 (9.57E-04) | 2.83E-01 – (5.51E-02, 12.2%) | 3.20E-01 ≈ (5.12E-04) | 8.46E-02 – (6.01E-02, 275.0%) | 3.20E-01 ≈ (1.10E-03) | 1.37E-01 – (9.61E-02, 132.0%) | **3.20E-01** ≈ (8.25E-04) | 1.69E-03 – (8.12E-04, 1.9E4%) | 3.19E-01 ≈ (8.12E-04) |
| ZDT3 | **5.08E-01** (9.81E-03) | 5.13E-01 (2.60E-04) | 5.07E-01 ≈ (1.33E-02, 0.2%) | 5.13E-01 ≈ (2.67E-04) | 4.03E-01 – (1.09E-01, 25.9%) | 5.13E-01 ≈ (6.82E-04) | 4.93E-01 – (2.18E-02, 3.0%) | 5.13E-01 ≈ (2.75E-04) | 2.77E-01 – (7.69E-04, 83.5%) | 5.12E-01 – (7.69E-04) |
| ZDT4 | **6.45E-01** (1.59E-02) | 6.54E-01 (7.22E-04) | 5.81E-01 – (6.57E-02, 10.9%) | **6.54E-01** ≈ (5.20E-04) | 3.85E-01 – (8.48E-02, 67.5%) | 6.53E-01 – (2.45E-03) | 2.01E-01 – (1.48E-01, 220.9%) | 6.54E-01 ≈ (1.31E-03) | 0.00E+00 – (2.52E-02, / ) | 6.30E-01 – (2.52E-02) |
| ZDT6 | **3.89E-01** (2.72E-03) | 3.90E-01 (1.06E-03) | 3.43E-01 – (5.47E-02, 13.3%) | **3.91E-01** ≈ (8.41E-04) | 2.72E-01 – (1.19E-01, 42.9%) | 3.90E-01 ≈ (1.36E-03) | 2.50E-01 – (1.36E-01, 55.7%) | 3.90E-01 ≈ (7.50E-04) | 0.00E+00 – (2.16E-03, / ) | 3.81E-01 – (2.16E-03) |
| DTLZ1 | 0.00E+00 (0.00E+00) | **3.39E-01** (3.35E-01) | 0.00E+00 ≈ (0.00E+00, / ) | 2.63E-01 ≈ (3.17E-01) | 0.00E+00 ≈ (0.00E+00, / ) | 1.01E-01 – (2.22E-01) | 0.00E+00 ≈ (0.00E+00, / ) | 1.62E-01 – (2.70E-01) | 0.00E+00 ≈ (3.16E-01, / ) | 3.27E-01 ≈ (3.16E-01) |
| DTLZ2 | 2.79E-01 (1.74E-02) | 3.36E-01 (8.43E-03) | 2.70E-01 ≈ (2.10E-02, 3.2%) | 3.37E-01 ≈ (9.31E-03) | 2.57E-01 – (2.27E-02, 8.5%) | 3.34E-01 ≈ (1.20E-02) | 2.48E-01 – (2.15E-02, 12.3%) | **3.37E-01** ≈ (1.29E-02) | **3.08E-01** + (1.23E-02, -9.3%) | 3.31E-01 ≈ (1.23E-02) |
| DTLZ3 | 0.00E+00 (0.00E+00) | 0.00E+00 (0.00E+00) | 0.00E+00 ≈ (0.00E+00, / ) | 0.00E+00 ≈ (0.00E+00) | 0.00E+00 ≈ (0.00E+00, / ) | 0.00E+00 ≈ (0.00E+00) | 0.00E+00 ≈ (0.00E+00, / ) | 0.00E+00 ≈ (0.00E+00) | 0.00E+00 ≈ (0.00E+00, / ) | 0.00E+00 ≈ (0.00E+00) |
| DTLZ4 | **2.81E-01** (1.73E-02) | **3.37E-01** (1.20E-02) | 2.68E-01 ≈ (6.99E-02, 4.7%) | 3.36E-01 ≈ (9.95E-03) | 2.50E-01 ≈ (5.07E-02, 12.3%) | 3.34E-01 ≈ (9.34E-03) | 2.11E-01 – (6.28E-02, 33.1%) | 3.34E-01 ≈ (9.97E-03) | 2.42E-01 ≈ (7.57E-02, 16.1%) | 3.19E-01 ≈ (7.57E-02) |
| DTLZ5 | 7.82E-02 (2.58E-03) | 8.86E-02 (5.66E-04) | 7.76E-02 ≈ (4.01E-03, 0.8%) | 8.90E-02 + (6.86E-04) | 7.20E-02 – (6.53E-03, 8.7%) | 8.89E-02 ≈ (4.24E-04) | 6.85E-02 – (6.84E-03, 14.2%) | 8.88E-02 ≈ (4.15E-04) | **8.28E-02** + (3.50E-04, -5.5%) | **8.93E-02** + (3.50E-04) |
| DTLZ6 | **8.14E-02** (1.03E-02) | 9.04E-02 (3.17E-04) | 1.74E-02 – (3.00E-02, 367.6%) | 9.03E-02 ≈ (3.48E-04) | 1.53E-02 – (2.39E-02, 433.2%) | **9.12E-02** + (4.33E-04) | 2.30E-03 – (1.03E-02, 3436.9%) | 9.04E-02 ≈ (3.61E-04) | 0.00E+00 – (0.00E+00, / ) | 0.00E+00 – (0.00E+00) |
| DTLZ7 | **2.54E-01** (6.67E-03) | **2.57E-01** (8.89E-03) | 2.50E-01 ≈ (1.32E-02, 1.4%) | 2.56E-01 ≈ (6.64E-03) | 2.27E-01 – (1.86E-02, 11.7%) | 2.53E-01 ≈ (8.68E-03) | 2.24E-01 – (1.51E-02, 13.1%) | 2.56E-01 ≈ (5.53E-03) | 1.41E-02 – (7.73E-03, 1701.8%) | 2.49E-01 – (7.73E-03) |
| Mean | 2.90E-01 (7.21E-03) | 3.31E-01 (3.07E-02) | 2.71E-01 (2.85E-02) | 3.25E-01 (2.89E-02) | 2.07E-01 (4.77E-02) | 3.11E-01 (2.16E-02) | 2.06E-01 (4.47E-02) | 3.16E-01 (2.53E-02) | 1.01E-01 (1.43E-03) | 3.18E-01 (4.41E-01) |

TABLE III
SUMMARIZED COMPARISON RESULTS AMONG SPEA2-KA, SPEA2-LA, SPEA2-M1/M2 AND SPEA2 BASED ON HV ON ZDT/DTLZ MOPS (EACH TUPLE *w/t/l* DENOTES THE ALGORITHM AT THE CORRESPONDING ROW WINS ON *w* MOPS, TIES ON *t* MOPS AND LOSES ON *l* MOPS, WHEN COMPARED TO THE ALGORITHM AT THE CORRESPONDING COLUMN).

| | SPEA2-KA | SPEA2-LA | SPEA2-M1 | SPEA2-M2 | SPEA2 |
|---|---|---|---|---|---|
| SPEA2-KA | – | 9/3/0 \| 1/10/1 | 10/2/0 \| 1/11/0 | 10/2/0 \| 2/10/0 | 7/4/1 \| 7/3/2 |
| SPEA2-LA | | – | 5/6/1 \| 2/9/1 | 7/5/0 \| 2/9/1 | 7/2/3 \| 7/3/2 |
| SPEA2-M1 | | | – | 2/10/0 \| 1/10/1 | 6/4/2 \| 7/2/3 |
| SPEA2-M2 | | | | – | 6/4/2 \| 7/2/3 |

TABLE IV
SUMMARIZED COMPARISON RESULTS AMONG NSGAII-KA, NSGAII-LA, NSGAII-M1/M2 AND NSGAII BASED ON HV ON ZDT/DTLZ MOPS (EACH TUPLE W/T/L DENOTES THE ALGORITHM AT THE CORRESPONDING ROW WINS ON W MOPS, TIES ON T MOPS AND LOSES ON L MOPS, WHEN COMPARED TO THE ALGORITHM AT THE CORRESPONDING COLUMN).

| | NSGAII-KA | NSGAII-LA | NSGAII-M1 | NSGAII-M2 | NSGAII |
|---|---|---|---|---|---|
| NSGAII-KA | – | 4/8/0 \| 0/11/1 | 9/3/0 \| 2/9/1 | 10/2/0 \| 1/11/0 | 7/3/2 \| 6/5/1 |
| NSGAII-LA | | – | 7/5/0 \| 3/8/1 | 9/3/0 \| 1/11/0 | 7/3/2 \| 7/5/0 |
| NSGAII-M1 | | | – | 3/8/1 \| 1/11/0 | 7/3/2 \| 6/4/2 |
| NSGAII-M2 | | | | – | 6/4/2 \| 7/3/2 |

search experiences provided by ZDT and DTLZ problems, respectively. In contrast, when one problem is being addressed by A-KA and A-LA, the rest 11 problems are all served as source problems that provide past search experiences to accelerate the optimization of the target problem. In spite of the transfer scheme, the evolutionary operators and other parameters settings are kept the same in the compared algorithms, which are configured based on [35] and summarized as follows.

1) Population size: $NP$ = 50.
2) Maximum function evaluation: *Max_evals* = 15000.
3) Independent run times: *runs* = 20.
4) Evolutionary operators and parameters in NSGAII and SPEA2:
   a) SBX crossover: $p_c$ = 0.9, $\eta_c$ = 20.
   b) Polynomial mutation: $p_m$ = 1/$D$, $\eta_m$ = 20.
5) Interval of knowledge transfer across problems: $TG$ = 10.
6) Solutions transferred from each source problem: $NT$ = 10 for A-KA and A-LA, $NT$ = $NP$ for A-M1 and A-M2.
7) Kernel function utilized in A-KA: polynomial kernel $\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{\mathrm{T}}\mathbf{y} + b)^d$ with $b$ = 0.1 and $d$ = 5 [52].

### *B. Results and Discussions*

The averaged hypervolume (HV) values as well as the standard deviation obtained by the classical MO algorithms, i.e., NSGAII and SPEA2, and their variants with different transfer schemes on the 12 ZDT/DTLZ multiobjective problems (MOPs) across 20 independent runs with 2500th and 15000th function evaluations are tabulated in Tables I and II. Superior performance is highlighted in bold, and the Wilcoxon rank sum test with 95% confidence level is conducted on the experimental results. $\approx$, + and – denote the compared algorithm is statistically significantly similar, better, and worse than the proposed SPEA2(NGSAII)-KA, respectively. In addition, the speed-up ratios of the proposed SPEA2(NGSAII)-KA against other compared algorithms at 2500th evaluation are also presented along with the standard deviation in the parenthesis. Further, to provide an overview of the performance among the algorithms, Tables III and IV summarized the comparison between the baseline MO solvers and their counterparts in terms of HV. In the tables, each grid contains a pair of tuples, in which the left and right tuples indicate the comparison results obtained at the 2500th and 15000th function evaluation, respectively. In particular, each tuple *w/t/l* denotes the algorithm in the corresponding row wins on *w* MOPs, ties on *t* MOPs, and loses on *l* MOPs, when compared to the algorithm in the corresponding column.

As can be observed in Tables I and II, firstly, the algorithms with the knowledge learned from past search experiences across heterogeneous problems all achieved superior solution quality over the baseline MO solvers SPEA2 and NGSAII. For instance, while both SPEA2 and NGSAII can only obtain a HV value of 0 on DTLZ 6 throughout the optimization, the other algorithms achieved relatively high HV values at the early 2500th function evaluation. Totally, as depicted in Table III and IV, the algorithms with knowledge transfer gained significantly better or competitive results on more than 9 out of totally 12 MOPs against SPEA2 and NSGAII.

Secondly, although the same linear autoencoding method is deployed, SPEA2-LA and NSGAII-LA outperformed NSGAII-M1/M2 and SPEA2-M1/M2 on most MOPs. In particular, SPEA2-LA and NSGAII-LA obtained higher HV values on 5 and 7 MOPs against SPEA2-M1 and NSGAII-M1, while on 7 and 9 MOPs against SPEA2-M2 and NSGAII-M2. Next, with regard to the proposed kernelized autoencoding method, it is observed that SPEA2-KA and NSGAII-KA both demonstrated superior performance over SPEA2-LA and NSGAII-LA in term of convergence speed. For instance, SPEA2-KA achieved 38.1% and 525.2% speed-up over SPEA2-LA on ZDT2 and DTLZ6, while NSGAII-KA converged 12.2% and 367.6% faster than NSGAII-LA. Totally, SPEA2-KA and NSGAII-KA achieved significantly better HV on 9 and 4 MOPs while competitive on others at the 2500th function evaluation.

Further, the convergence trends on the 12 problems are presented in Figs. 4 and 5 to assess the efficiency of the compared methods. In particular, the convergence graphs of SPEA2-KA, SPEA2-LA, SPEA2-M1, SPEA2-M2 and SPEA2 are depicted in Fig. 4, while Fig. 5 shows the convergence graphs of NSGAII+KA, NSGAII+LA, NSGAII-M1, NSGAII-M2 and NSGAII. In these figures, the Y-axis denotes the mean hypervolume value obtained across 20 independent runs, while the X-axis denotes the number of fitness evaluations made so far. From Fig. 4, we have the following observations: (1) Algorithms with different knowledge transfer schemes converged faster than SPEA2 on 9 out of 12 MOPs; (2) SPEA2-KA and SPEA2-LA have better performance than SPEA2-M1 and SPEA2-M2 on 10 out of 12 MOPs. Representative examples are Fig. 4(b) and 4(k); (3) SPEA2-KA achieved enhanced convergence performance against SPEA2-LA, which owes to the proposed KA method. For instance, on ZDT2 (Fig. 4(b)), ZDT6 (Fig. 4(e)) and DTLZ6 (Fig. 4(k)), SPEA2-KA achieved a steep convergence trend that saved about 3000 function

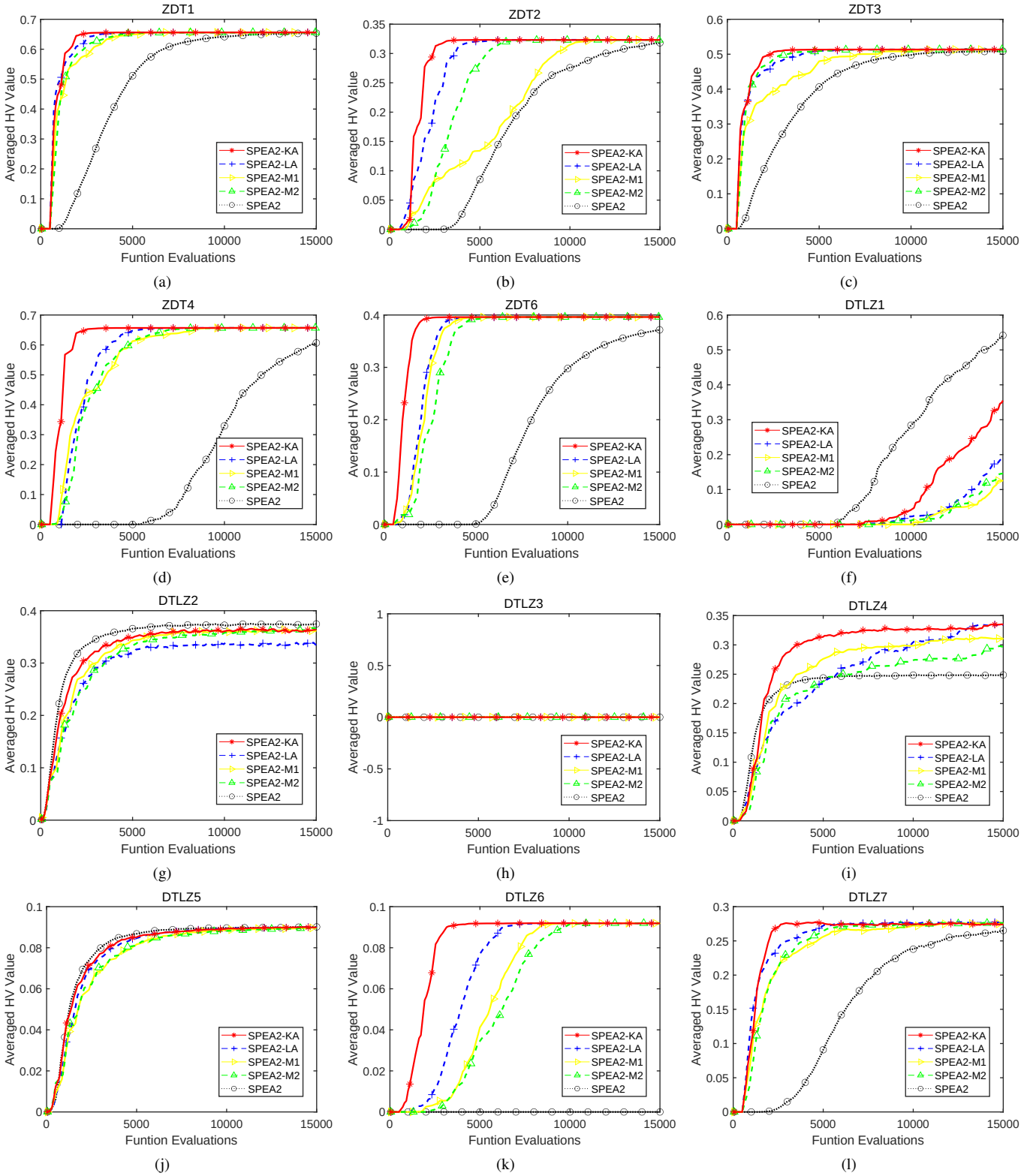



Fig. 4. The averaged Hypervolume curves obtained by SPEA2-KA, SPEA2-LA, SPEA2-M1, SPEA2-M2 and SPEA2 on the 12 ZDT/DTLZ multiobjective problems across 20 independent runs. y-axis: hypervolume , x-axis: function evaluations.

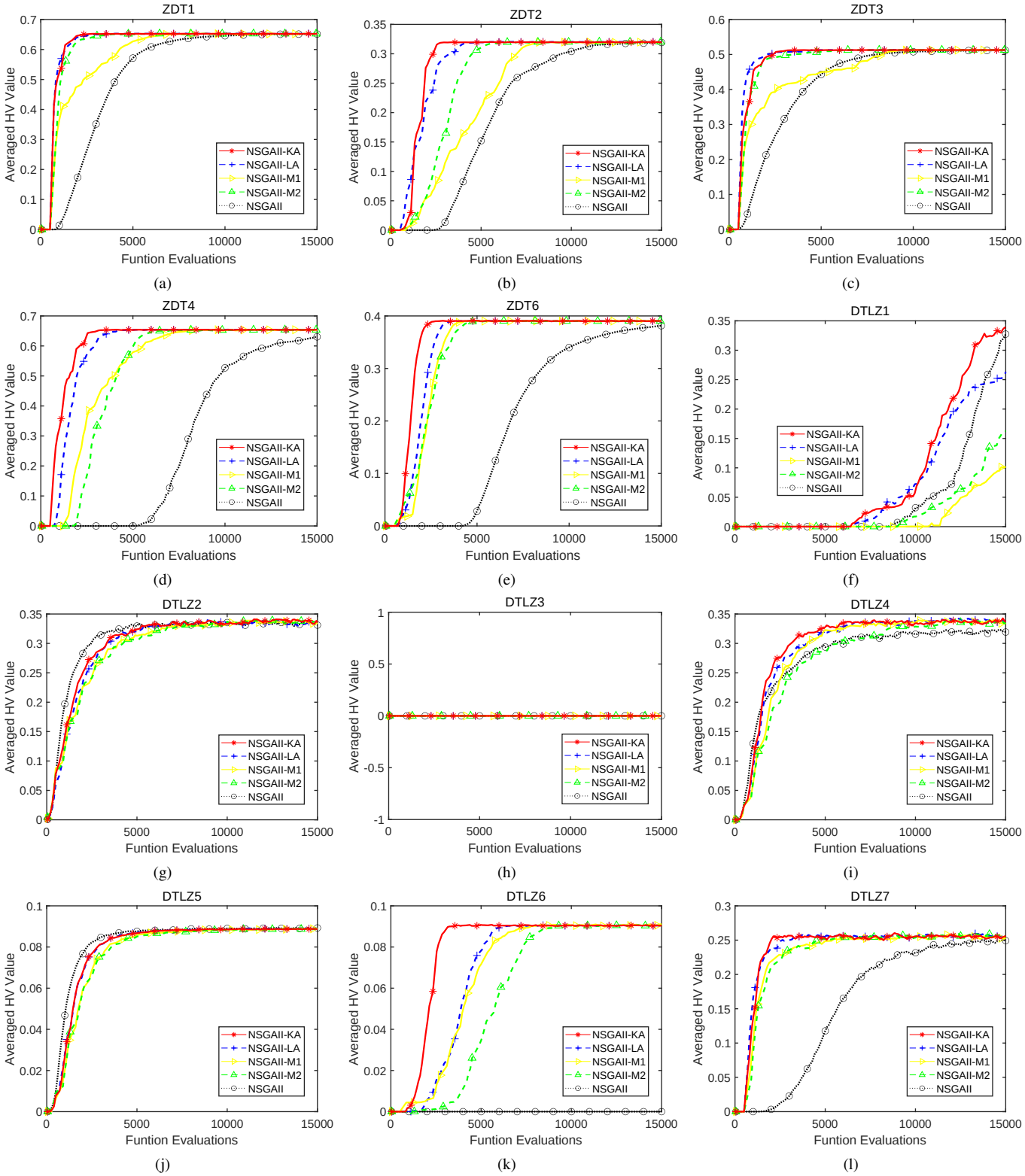



Fig. 5. The averaged Hypervolume curves obtained by NSGAII-KA, NSGAII-LA, NSGAII-M1, NSGAII-M2 and NSGAII on the 12 ZDT/DTLZ multiobjective problems across 20 independent runs. y-axis: hypervolume , x-axis: function evaluations.

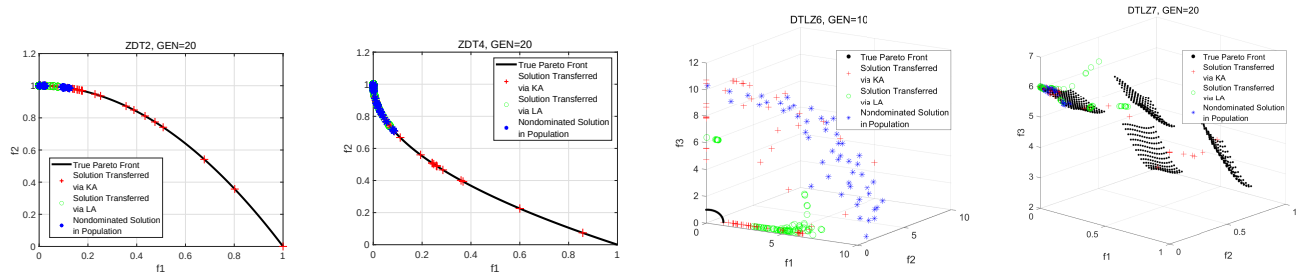



Fig. 6. Illustration of the transferred solutions obtained via KA and LA and the nondominated solutions in the population at early stage on representative multiobjective problems. GEN denotes the generation number.

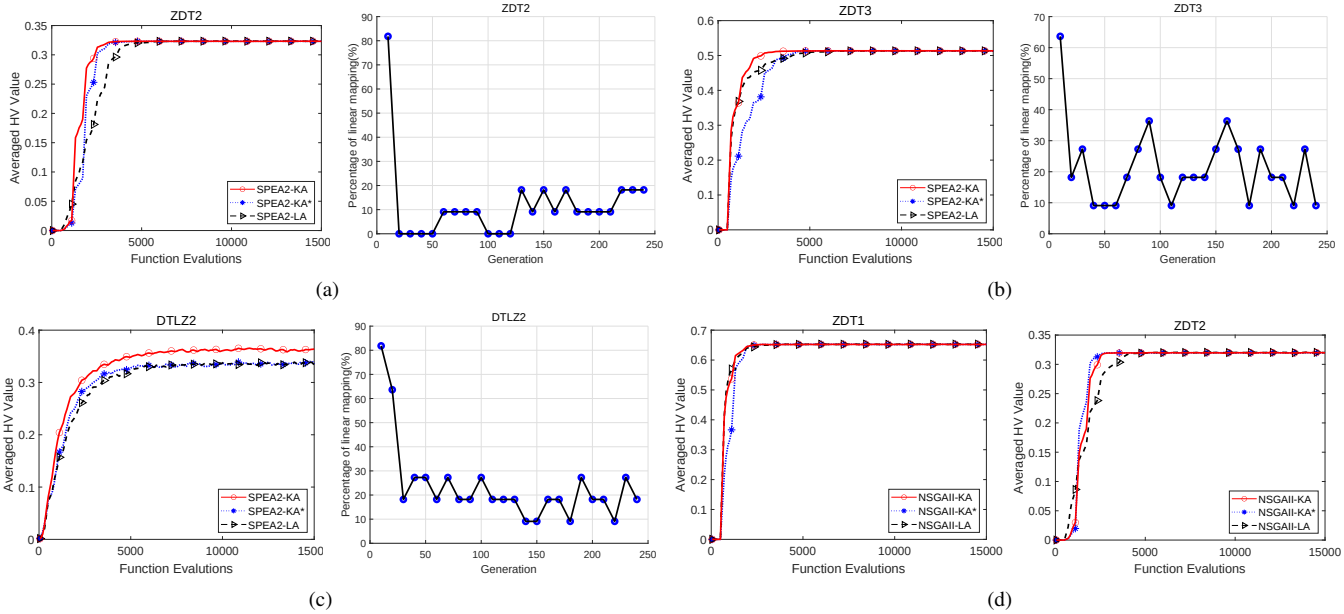



Fig. 7. Convergence graphs of SPEA2(NGSAII)-KA, SPEA2(NGSAII)-KA* and SPEA2(NGSAII)-LA as well as the percentage of linear mapping performed by SPEA2-KA in each transfer phase on representative ZDT/DTLZ problems. SPEA2(NGSAII)-KA adaptively executes linear or nonlinear mapping, while SPEA2(NGSAII)-KA* and SPEA2(NGSAII)-LA perform solely linear and nonlinear mapping, respectively.

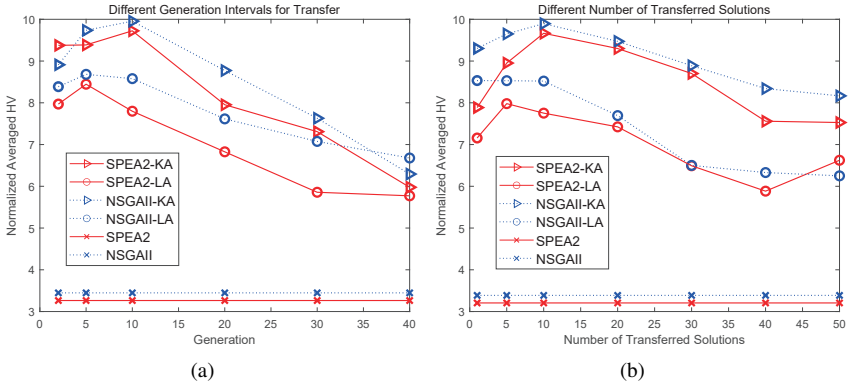



Fig. 8. Normalized average HV obtained by SPEA2(NGSAII)-KA against SPEA2(NGSAII)-LA and SPEA2(NGSAII) on all the 12 multi-objective benchmarks at 2500th function evaluation across 20 independent runs with various configurations of *TG* and *NT*.

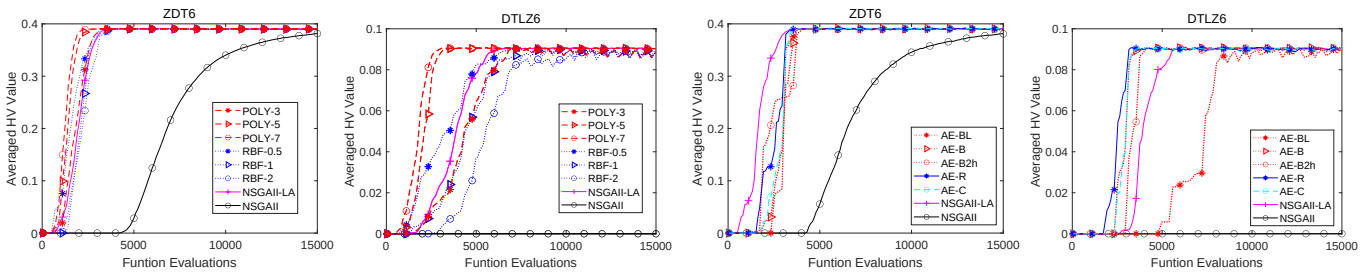



Fig. 9. (a) Convergence graphs of POLY-3, POLY-5, POLY-7, RBF-0.5, RBF-1, RBF-2 as well as NSGAII-LA and NSGAII across 20 independent runs on ZDT6 and DTLZ6. (b) Convergence graphs of AE-B, AE-C, AE-R, AE-BL, AE-B2h as well as NSGAII-LA and NSGAII across 20 independent runs on ZDT6 and DTLZ6.

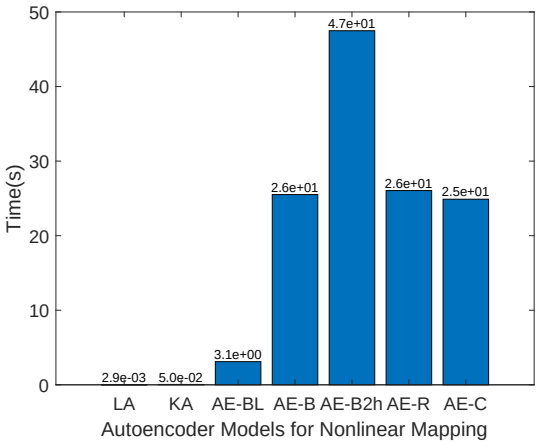



Fig. 10. The computational time consumed for training each of the seven autoencoder models.

evaluations to arrive at the HV value obtained by SPEA2-LA at around the 5000th evaluation. Similar observations have also been achieved in Fig. 5 when configuring NSGAII as the baseline MO solver.

Lastly, Fig. 6 illustrates the transferred solutions generated by the proposed kernelized autoencoding, the linear autoencoding and the non-dominated solutions in the population at early stage on representative MO problems. As can be observed from Fig. 6, the solutions obtained via KA (the red plus) not only locate close to the true Pareto front but also scatter better than those generated via LA (the green circle) and the non-dominated solutions in current population, which accounts for the superiority of SPEA2-KA and NSGAII-KA as shown in Figs. 4 and 5.

In summary, the following conclusions can be drawn from the experimental results above. (1) The performance of EA can be greatly improved by properly reusing the knowledge from related problem; (2) The kernelized autoencoding method leads to an enhanced evolutionary search in terms of both solution quality and convergence speed against the linear autoencoding method; (3) The proposed KAES works more efficiently than the AEES proposed in [35].

### *C. The efficacy of linear-nonlinear combination and adaptation*

To provide deeper insight of the performance obtained by the proposed KAES and demonstrate the efficacy of the combination and adaptation of linear and nonlinear mapping, SPEA2(NGSAII)-KA that adaptively executes linear or nonlinear mapping is compared with SPEA2(NGSAII)-KA* and SPEA2(NGSAII)-LA which perform solely nonlinear and linear mapping, respectively. Fig. 7 illustrates the convergence curves of SPEA2(NGSAII)-KA, SPEA2(NGSAII)-KA* and SPEA2(NGSAII)-LA on representative ZDT/DTLZ problems. Along with the convergence graph, the percentage of linear mapping performed by SPEA2-KA in each transfer phase is also presented in Fig. 7(a)-7(c).

From Fig. 7(a) and 7(b), the adaptation can be clearly observed. In particular, SPEA2-KA kept the percentage of linear mapping at a low level on ZDT2 where SPEA2-KA* converged faster than SPEA2-LA, while more linear mappings were performed on ZDT3 where SPEA2-LA outperformed SPEA2-KA*. As a result, SPEA2-KA achieved superior performance on both ZDT2 and ZDT3. Next, Fig. 7(c) indicated that the combination of linear and nonlinear mappings may sometimes further enhance optimization performance compared to conducting linear or nonlinear mapping independently. Similar adaptation can also be observed when NSGAII is configured as the baseline algorithm which is depicted in Fig. 7(d).

### *D. Parametric analysis*

In the proposed KAES, *TG* and *NT* are two important parameters that define the frequency and amount of knowledge sharing between tasks. We further investigate how the configurations of *TG* and *NT* affect the performance of the proposed KAES.

On one hand, a small value of *TG* and a big value of NT will increase the frequency and amount of knowledge sharing across tasks which can make better use of the available knowledge. On the other hand, excessive knowledge transfer would also consume a large number of additional function evaluations, which may greatly deteriorate the optimization performance. Fig. 8 illustrates the normalized average HV obtained by SPEA2(NGSAII)-KA against SPEA2(NGSAII)-LA and SPEA2(NGSAII) on all the 12 multi-objective bench-

marks at 2500th function evaluation across 20 independent runs with various configurations of *TG* and *NT*. As can be observed, SPEA2(NGSAII)-KA and SPEA2(NGSAII)-LA with knowledge transfer achieved significantly better results than the standard SPEA2(NGSAII) with all the configurations of *TG* and *NT*. Further, SPEA2(NGSAII)-KA always outperform SPEA2(NGSAII)-LA under the same setting of *TG* and *NT*.

It is also noted that in Fig. 8(a) the averaged HV of SPEA2(NGSAII)-KA and SPEA2(NGSAII)-LA increased with *TG* at first since lots of functions evaluations are saved with the slowdown in transfer frequency. Then, the HV dropped continuously when *TG* was further increased because of the insufficient use of knowledge. Similar trends are also observed with different *NT* in Fig. 8(b)

Lastly, although the optimal configurations of *TG* and *NT* are generally problem-dependent, *TG*=10 and *NT*=10 is adopted as defaults since it provides stable yet superior results across most of the problems.

### *E. Kernel functions and autoencoder models*

In this section, we present the preliminary study to investigate the performance of the proposed KAES by employing different kernel functions and multilayer autoencoders for nonlinear mapping.

First of all, the commonly-used polynomial kernel $\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{\mathrm{T}}\mathbf{y} + 0.1)^d$ and radial basis function (rbf) kernel $\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\delta^2}\right)$ with different settings are compared. In particular, by setting $d$ as 3, 5 and 7 and $\delta$ as 0.5, 1 and 2, we obtained six kernel functions. We denote the NSGAII-KA variants equipped with these six kernels as POLY-3, POLY-5, POLY-7, RBF-0.5, RBF-1 and RBF-2. Fig. 9(a) presents the convergence graphs of the NSGAII-KA variants as well as NSGAII-LA and NSGAII on ZDT6 and DTLZ6, respectively. It can be observed that the kernel functions exhibit different convergence speed. In general, polynomial kernel converged faster than rbf kernel. Furthermore, different settings of the same kernel function also greatly affect the performance. For instance, while POLY-5 and POLY-7 significantly outperformed NSGAII-LA on DTLZ6, POLY-3 was slightly inferior to NSGAII-LA. As can been seen, the optimal kernel as well as its best configuration is usually problem-dependent. In this work, we took POLY-5, i.e., $\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{\mathrm{T}}\mathbf{y} + 0.1)^5$, which exhibited reasonable performance on most of the MOPs as the default kernel in the experiments.

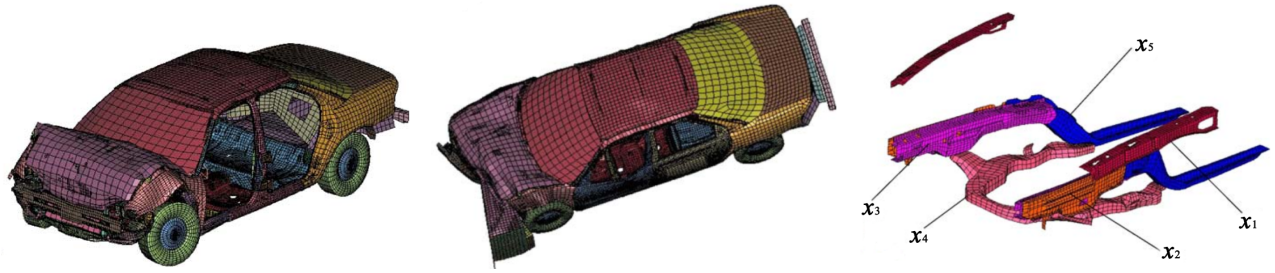
Next, we study on three typical autoencoder models, i.e., the basic autoencoder (denoted as AE-B), contractive autoencoder (denoted as AE-C) and robust autoencoder (denoted as AE-R), for nonlinear mapping. For brevity, the readers are referred to [48] for more details of these AE models. The three models are all configured with one hidden layer of 50 neurons and ReLu activation function, which are trained over 10000 iterations to build the mapping. In addition, two variants of AE-B, one of which is trained over 1000 iterations (denoted as AE-BL) while the other has two hidden layers (denoted as AE-B2h), are also compared. The convergence graphs of the five models as well as NSGAII-LA and NSGAII on ZDT6 and DTLZ6 are shown in Fig. 9(a).

From the results, we have the following observations (1) AE-R, AE-C and AE-B2h performed similarly with the basic AE-B on both ZDT6 and DTLZ6. This is because the strength of these advanced models cannot be revealed on a small dataset, i.e., a population of 50 solutions. (2) While most AE models outperformed NSGAII-LA on DTLZ6, they are all inferior to NSGAII-LA on ZDT6. As the performance of the AE model can be affected by many factors such as architecture, it makes the use of AE models for nonlinear mapping not as stable as KA and LA. (3) AE-B was much better than AE-BL trained by only 1000 iterations. Hence, it is expected that by increasing the training iterations, the performance can be further improved. Nevertheless, it must be pointed out that the training of these models is a time-consuming process. Fig. 10 summarizes the computational time spent for training each of the seven autoencoder models. It is observed that even AE-BL takes about 3 seconds in training. As the mapping is built on each pair of source and target problem and is constantly updated, it is estimated that AE-BL needs about 15 minutes for the mapping construction across the optimization under the current setting. In contrast, both KA and LA consume less than 0.1 second to build a mapping, which will not bring much computational burden in the search process. The above observations indicate that the multilayer AE models are not suitable for nonlinear mapping in the proposed KAES.

## V. Real-world Case Study on the Vehicle Crashworthiness Design Problem

Vehicle crashworthiness is one of the most important design considerations in the automotive industry, which is directly and closely related to the occupant safety. The goal of crashworthiness design is to work out a vehicle structure that can largely absorb the crash energy by controlled vehicle deformations while maintaining adequate space for the occupant [53]. Normally, the vehicle structure is firstly simulated and optimized on computer to meet various safety guidelines before putting into production. However, the simulations are usually computationally expensive such that even a single evaluation may take several minutes. In practice, the same type of vehicles often share similar structures, which makes it possible to leverage the large amount of past experience to guide the crashworthiness design of a new vehicle model. In this section, we present a case study on the vehicle crashworthiness design problem to show the benefit of knowledge transfer for the design process and validate the proposed kerelized autoencoding evolutionary search.

A full vehicle crashworthiness design includes the simulation of a variety of crash events, such as roof crush, interior squash and frontal, side and rear impacts [53]. According to [39], in this study, we consider a full-width frontal crash test and an offset-frontal crash test. Figs. 11(a) and 11(b) illustrates the full-width frontal crash and the offset-frontal crash, respectively. Particularly, the offset-frontal test demands a high degree of vehicle stiffness to resist intrusion while the frontal test requires a relatively low degree of stiffness to alleviate the injuries caused by deceleration [54]. To improve




(a) (b) (c)

Fig. 11. (a) Illustration of the full-width frontal crash; (b) Illustration of the offset-frontal crash; (c) Illustration of the five decision variables of the multiobjective problem [39].

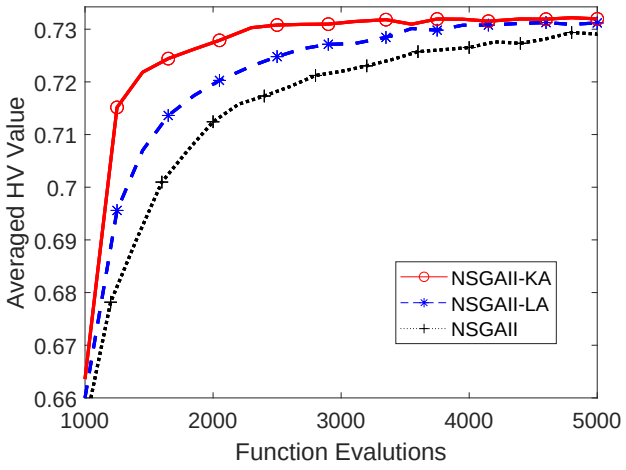



Fig. 12. The averaged Hypervolume curves obtained by NSGAII-KA and NSGAII-LA and NSGAII across 20 independent runs. y-axis: hypervolume , x-axis: function evaluations.

the overall crashworthiness of the frontal structure of a vehicle, the two tests thus should be considered simultaneously as a multiobjective problem, which can be formulated as follow:

$$\begin{aligned} min\ \mathbf{F}(\mathbf{x}) &= (f_{mass}, f_{A_{in}}, f_{Intrusion}) \\ s.t.\ \mathbf{x} &= (x_1, x_2, x_3, x_4, x_5) \in [1,3]^5, \end{aligned} \tag{11}$$

where $(x_1, x_2, x_3, x_4, x_5)$ denote the 5 design variables, each of which corresponds to the thickness of a particular reinforced member around the frontal structure as shown in Fig. 11(c). $f_{mass}, f_{A_{in}}, f_{Intrusion}$ represent the 3 objectives, which are the mass of the vehicle, the collision acceleration in the "full frontal crash" and the toe board intrusion in the "offset-frontal crash". The simulation on a National Highway Transportation and Safety Association vehicle as described in [39] is considered here as the target problem. Further, we adjust some design criteria to construct 5 distinct but related models and solve them in advance so that the obtained design experience can be reused to accelerate the optimization of the target problem. For a more detailed description about the problem formulation and the setting of the simulation, the reader is referred to [39].

The averaged convergence trends of HV value obtained by the NSGAII-KA and NSGAII-LA and NSGAII across 20 independent runs are presented in Fig. 12. As can be clearly observed from Fig. 12, the algorithms with knowledge transfer, i.e., NSGAII-KA and NSGAII-LA, achieved superior performance in terms of both solution quality and search speed against the classical NSGAII, which demonstrates the effectiveness of knowledge transfer for enhanced evolutionary search. On the other hand, NSGAII-KA outperformed NSGAII-LA and NSGAII all along the search process. To demonstrate, it only takes NSGAII-KA around 2000 and 3000 function evaluations, respectively, to reach the same HV value obtained by NSGAII-LA and NSGAII at the 5000th evaluation. The kernelized autoencoding method boosts the optimizer to quickly achieve good solutions with a lower number of function evaluations, which can thus significantly shorten the design cycle.

From a broader perspective, on one hand many practical industrial systems are expected to tackle a large number of related problems over their lifetime where a vast pool of knowledge can be reserved for further reuse [22]. On the other hand, it is the urgent demands in present industry to achieve high-quality solutions within strict time constraints. Consequently, the proposed KAES that is able to enhance the optimization performance by effectively exploiting the useful knowledge learned from past experiences has a great potential to be applied to the solving of a variety of real-world optimization problems.

## VI. Conclusion

In this paper, a new learnable evolutionary search paradigm, namely kernelized autoencoding evolutionary search (KAES), has been proposed. In KAES, a kernelized autoencoder with a closed-form solution has been derived to construct the nonlinear mapping across heterogeneous problems in a Reproducing Kernel Hilbert Space (RKHS). The proposed KAES adaptively selects the linear or kernelized autoencoder to build the inter-task mapping along the search process. With the learned mapping, the knowledge from past search experience can be learned and transferred to the current problem in the form of adapted problem solutions. Furthermore, in view of the mapping effectiveness and computational efficiency, we have proposed to sort the solutions used in the mapping construction before kernalized autoencoding process to increase the ordinal correlation and transfer a limited number rather than the whole population of the past optimized solutions. To validate the efficacy of the proposed KAES, comprehensive empirical studies have been conducted on 12 complex multiobjectives

benchmarks and a real-world application on the vehicle crashworthiness design. The obtained results demonstrated that the proposed KAES is able to enhance the evolutionary search in terms of both solution quality and search speed when compared to the state-of-art AEES as well as the classical MO algorithms.

For future work, we would like to embark on an in-depth study to analyze how different the kernel functions affect the performance of KAES on optimization problems with different characteristics and explore the efficient way to exploit the state-of-art autoencoders for nonlinear mapping. We would also further expand the applicability of our approach to a wider variety of real-world optimization problems.

## VII. Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61876025, in part by the Venture & Innovation Support Program for Chongqing Overseas Returnees under Grant No. cx2018044 and cx2019020, and in part by the A*STAR Cyber-Physical Production System (CPPS) – Towards Contextual and Intelligent Response Research Program, under the RIE2020 IAF-PP Grant A19C1a0018.

## References

[1] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 3–17, 1997.

[2] K. Deb, "Genetic algorithm in search and optimization: the technique and applications," in *Proceedings of international workshop on soft computing and intelligent systems*, 1998, pp. 58–87.

[3] H. Maaranen, K. Miettinen, and A. Penttinen, "On initial populations of a genetic algorithm for continuous optimization problems," *Journal of Global Optimization*, vol. 37, no. 3, p. 405, 2007.

[4] J. Sun, Q. Zhang, and E. P. Tsang, "De/eda: A new evolutionary algorithm for global optimization," *Information Sciences*, vol. 169, no. 3-4, pp. 249–262, 2005.

[5] J. Zhang, V. Avasarala, A. C. Sanderson, and T. Mullen, "Differential evolution for discrete optimization: An experimental study on combinatorial auction problems," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 2008, pp. 2794–2800.

[6] T. P. Dinh, B. H. T. Thanh, T. T. Ba, and L. N. Binh, "Multifactorial evolutionary algorithm for solving clustered tree problems: competition among cayley codes," *Memetic Computing*, vol. 12, no. 3, pp. 185–217, 2020.

[7] Y. Wang and C. Dang, "An evolutionary algorithm for global optimization based on level-set evolution and latin squares," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 579–595, 2007.

[8] L. Deng, L. Zhang, H. Sun, and L. Qiao, "DSM-DE: a differential evolution with dynamic speciation-based mutation for single-objective optimization," *Memetic Computing*, vol. 12, no. 1, pp. 73–86, 2020.

[9] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[10] S. Jiang and S. Yang, "An improved multiobjective optimization evolutionary algorithm based on decomposition for complex pareto fronts," *IEEE transactions on Cybernetics*, vol. 46, no. 2, pp. 421–437, 2015.

[11] S. Jiang and S. Yang, "A strength pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 329–346, 2017.

[12] Z. He, G. G. Yen, and Z. Yi, "Robust multiobjective optimization via evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 316–330, 2018.

[13] Z. He, G. G. Yen, and J. Lv, "Evolutionary multi-objective optimization with robustness enhancement," *IEEE Transactions on Evolutionary Computation*, 2019.

[14] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[15] S. Jiang and S. Yang, "A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 65–82, 2016.

[16] K. Deb, R. B. Agrawal *et al.*, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.

[17] A. V. Eremeev and Y. V. Kovalenko, "A memetic algorithm with optimal recombination for the asymmetric travelling salesman problem," *Memetic Computing*, vol. 12, no. 1, pp. 23–36, 2020.

[18] C.-K. Ting, C.-F. Ko, and C.-H. Huang, "Selecting survivors in genetic algorithm using tabu search strategies," *Memetic Computing*, vol. 1, no. 3, pp. 191–203, 2009.

[19] S. Yang, M. Li, X. Liu, and J. Zheng, "A grid-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 721–736, 2013.

[20] Y. S. Ong and A. J. Keane, "Meta-lamarckian learning in memetic algorithms," *IEEE transactions on evolutionary computation*, vol. 8, no. 2, pp. 99–110, 2004.

[21] A. Gupta and Y.-S. Ong, *Memetic computation: the mainspring of knowledge transfer in a data-driven optimization era*. Springer, 2018, vol. 21.

[22] A. Gupta, Y.-S. Ong, and L. Feng, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 51–64, 2017.

[23] Y.-S. Ong and A. Gupta, "Air 5: Five pillars of artificial intelligence research," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 5, pp. 411–415, 2019.

[24] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 193–200.

[25] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[26] S. Louis and J. McDonnell, "Learning with case-injected genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 4, pp. 316–328, 2004.

[27] S. J. Louis and G. Li, "Case injected genetic algorithms for traveling salesman problems," *Information sciences*, vol. 122, no. 2-4, pp. 201–225, 2000.

[28] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 3, 1999, pp. 1875–1882.

[29] G. J. Barlow and S. F. Smith, "A memory enhanced evolutionary algorithm for dynamic scheduling problems," in *Workshops on Applications of Evolutionary Computation*, 2008, pp. 606–615.

[30] S. Yang, "Explicit memory schemes for evolutionary algorithms in dynamic environments," *Evolutionary Computation in Dynamic and Uncertain Environments*, pp. 3–28, 2007.

[31] L. Feng, Y.-S. Ong, M.-H. Lim, and I. W. Tsang, "Memetic search with interdomain learning: A realization between cvrp and carp," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 644–658, 2014.

[32] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2016.

[33] A. T. W. Min, Y.-S. Ong, A. Gupta, and C.-K. Goh, "Multiproblem surrogates: transfer evolutionary multiobjective optimization of computationally expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 15–28, 2017.

[34] B. Da, A. Gupta, and Y.-S. Ong, "Curbing negative influences online for seamless transfer evolutionary optimization," *IEEE Transactions on Cybernetics*, 2018.

[35] L. Feng, Y.-S. Ong, S. Jiang, and A. Gupta, "Autoencoding evolutionary search with learning across heterogeneous problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 760–772, 2017.

[36] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

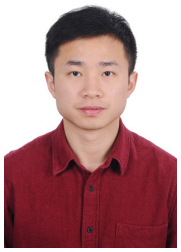
[37] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[38] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary multiobjective optimization*, 2005, pp. 105–145.

[39] X. Liao, Q. Li, X. Yang, W. Zhang, and W. Li, "Multiobjective optimization for crash safety design of vehicles using stepwise regression model," *Structural and multidisciplinary optimization*, vol. 35, no. 6, pp. 561–569, 2008.
[40] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.
[41] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th international conference on machine learning*, 2011, pp. 513–520.
[42] M. Chen, K. Q. Weinberger, Z. Xu, and F. Sha, "Marginalizing stacked linear denoising autoencoders," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 3849–3875, 2015.
[43] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder." in *Interspeech*, 2013, pp. 436–440.
[44] T. Ishii, H. Komiyama, T. Shinozaki, Y. Horiuchi, and S. Kuroiwa, "Reverberant speech recognition based on denoising autoencoder." in *Interspeech*, 2013, pp. 3512–3516.
[45] H. Sagha, N. Cummins, and B. Schuller, "Stacked denoising autoencoders for sentiment analysis: a review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 5, p. e1212, 2017.
[46] S. Zhai and Z. M. Zhang, "Semisupervised autoencoder for sentiment analysis," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
[47] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
[48] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Information Fusion*, vol. 44, pp. 78–96, 2018.
[49] X. He and P. Niyogi, "Locality preserving projections," in *Advances in neural information processing systems*, 2004, pp. 153–160.
[50] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.
[51] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.
[52] G. Valentini and T. G. Dietterich, "Bias-variance analysis of support vector machines for the development of svm-based ensemble methods," *Journal of Machine Learning Research*, vol. 5, no. Jul, pp. 725–775, 2004.
[53] P. Du Bois, C. C. Chou, B. B. Fileta, T. B. Khalil, A. I. King, H. F. Mahmood, H. J. Mertz, J. Wismans, P. Prasad, and J. E. Belwafa, "Vehicle crashworthiness and occupant protection," 2004.
[54] J. Xiaochun and W. Xiao, "Simulation of crashworthiness during front impact and offset impact and vehicle body structure improvement," *J Automotive Safety and Energy*, vol. 2, no. 3, pp. 212–216, 2011.

**Lei Zhou** received the B.E. degree from the School of Computer Science and Technology, Shandong University, Shandong, China, in 2014, and the Ph.D. degree from the College of Computer Science, Chongqing University, Chongqing, China, in 2019.

He is currently a Post-Doctoral Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. His current research interests include evolutionary computations, memetic computing, as well as transfer learning and optimization.



**Liang Feng** received the Ph.D degree from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2014. He was a Postdoctoral Research Fellow at the Computational Intelligence Graduate Lab, Nanyang Technological University, Singapore. He is currently a Professor at the College of Computer Science, Chongqing University, China. His research interests include Computational and Artificial Intelligence, Memetic Computing, Big Data Optimization and Learning, as well as Transfer Learning. His research work on evolutionary multitasking won the 2019 IEEE Transactions on Evolutionary Computation Outstanding Paper Award. He is Associate Editor of the IEEE Transactions on Emerging Topics in Computational Intelligence, IEEE Computational Intelligence Magazine, Memetic Computing, and Cognitive Computation. He is also the founding Chair of the IEEE CIS Intelligent Systems Applications Technical Committee Task Force on Transfer Learning & Transfer Optimization.

**Abhishek Gupta** received his PhD in Engineering Science from the University of Auckland, New Zealand, in the year 2014. He currently serves as a Scientist and Technical Lead in the Singapore Institute of Manufacturing Technology (SIMTech), at the Agency for Science, Technology and Research (A*STAR), Singapore. He has diverse research experience in the field of computational science, ranging from numerical methods in engineering physics, to topics in computational intelligence. His current research interests lie in probabilistic model-based search algorithms with transfer and multitask learning capabilities, for applications spanning cyber physical production systems and engineering design.

**YEW-SOON ONG (M'99-SM'12-F'18)** received the Ph.D. degree in artificial intelligence in complex design from the University of Southampton, U.K., in 2003. He is Presidents Chair Professor in Computer Science at Nanyang Technological University (NTU), and holds the position of Chief Artificial Intelligence Scientist of the Agency for Science, Technology and Research Singapore. At NTU, he serves as Director of the Data Science and Artificial Intelligence Research and co-Director of the Singtel-NTU Cognitive & Artificial Intelligence Joint Lab. His research interest is in artificial and computational intelligence. He is founding Editor-in-Chief of the IEEE Transactions on Emerging Topics in Computational Intelligence and AE of IEEE Transactions on Neural Networks & Learning Systems, IEEE on Transactions on Cybernetics, IEEE Transactions on Artificial Intelligence and others. He has received several IEEE outstanding paper awards and was listed as a Thomson Reuters highly cited researcher and among the World's Most Influential Scientific Minds.